# 古くて新しい代数方程式の解法(科学技術における数値計算の理論と応用)

AUTHOR(S):

中島, 勝也; 千葉, 芳之

CITATION:

中島, 勝也 ...[et al]. 古くて新しい代数方程式の解法(科学技術における数値計算の理論と応用). 数理解析研究所講究録 1996, 944: 199-207

ISSUE DATE:

1996-04

URL:

http://hdl.handle.net/2433/60183

数理解析研究所講究録
944巻 1996年 199-207



# 古くて新しい代数方程式の解法

早稲田大学理工学部教授　中島 勝也(Katsuya Nakashima)
早稲田大学大学院理工学研究科　千葉 芳之(Yoshiyuki Chiba)

## 1．問題の発端

数学ソフトウェア「*Mathematica*」を用いて、ランダムに係数を与えた50次の代数方程式を解いて、その近似解における関数値の最大絶対値の誤差を調べたところ、その正確さと計算時間の速さに驚いた事が、その問題について深く考えさせられた契機となった。

## 2．従来の方法

計算機出現以来、Newton法、複素Newton法、２次元Newton法であるBairstow-Hitchcock法、山本哲朗氏が推奨したＤＫＡ法が主に用いられていて、教科書的例題では高々20次ぐらいの低次方程式についての解説に限られていた。

Newton法においては高次の反復式を利用するので、初期値の設定がその収束に大きく影響し、カオスやフラクタルなど収束しないケースもあり、ジュリア集合やマンデルブロー集合など、副次的話題も起きたが、速く根を求めたい要求には応えていない。

## 3．我々の提案

50次の代数方程式でも１分以内に、すべての解を求めている*Mathematica*の関数NSolveの解法については利用者には分からないので、ここに提案する方法が、すでに*Mathematica*で用いられている解法と同一であるかどうかは分からない。我々は、Newton反復に表れるカオスをさけるためにNewton法以外の解法を選ぶことにした。以下にその概要を述べる。

Ⅰ）与えられる多項式 $f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$ の各係数 $a_i$ は有理数とし、はじめにの重根を除く。$f(x)$ と $f'(x)$ との共通因数gcdを求めて、$f(x)/gcd$ を $f_0(x)$ とおく。

Ⅱ）有界区間内の $f(x)=0$ の実根の個数をスツルムの定理により調べ、実根が１個づつ存在する区間に分割する。その区間において２分割法により実根を確定する。

Ⅲ）$f(x)=0$ の虚根を求める。$f(a+bi)=u(a,b)+b\ v(a,b)i$ とおき、$u(a,b)=0$, $v(a,b)=0$ からｂを消去する。その結果の方程式からａの実根を解く。その実根ａを $u(a,b)=0$, $v(a,b)=0$ に代入して両者を満たす正根ｂを求める。$a\pm bi$ を $f(x)=0$ の２根の組とする。

## 4．*Mathematica*の組み込みオブジェクトNSolveによる解計算

方程式 $x+2x^2+3x^3+\cdots+50x^{50}=0$ を解いたときの最大誤差は次の通りである。

```
f[x_]:=Sum[ i (x^i) , {i,1,50} ]
Max[ Abs[ f[x]/.NSolve[f[x]==0,x] ] ]//Timing
                                  -7
{1.86 Second, 9.24846 10   }
```

方程式 $x+2x^2+3x^3+\cdots+8x^8=0$ を解いたときの最大誤差は次の通りである。

```
f[x_]:=Sum[ i (x^i) , {i,1,8} ]
Max[ Abs[ f[x]/.NSolve[f[x]==0,x] ] ]//Timing
                                   -15
{0.11 Second, 1.3552 10    }
```

☆Sum[*f*,{*i*,*imin*,*imax*}]：総和 $\sum_{imin}^{imax} f$ を評価する。
☆Max[$x_1$,$x_2$,...]：$x_i$の内でその数値が最大のものを返す。
☆Abs[*z*]：実数及び複素数*z*の絶対値を求める。
☆*expr*/.*rules*：*rules*を*expr*に適用した結果を与える。
☆NSolve[*lhs*==*rls*,*var*]：多項式方程式の解の数値的な近似のリストを与える。
☆Timing[*expr*]：*expr*を評価して、この結果と所要時間のリストを返す。
（*expr*//Timingは Timing[*expr*]の簡略形）

## 5．我々の方法による解法

方程式 $x+2x^2+3x^3+\cdots+8x^8=0$ を解く。そのために用いるプログラムを以下に記す。

●スツルム列作成。

(1) ｓｔｕｒｍ1

引数を関数とし、その関数の重根を取り除き、スツルム列$f_0(x), f_1(x)$を作成するものである。

$f_0(x)=f(x)/\mathrm{gcd}$ とした。

☆Dt[*f*,*x*]： 全微分 $\frac{d}{dx}f$ を与える。

☆Simplify[*expr*]：*expr*に対して代数的な変換のシーケンスを実行し、それが見いだす最も簡素な形式を返す。

☆PolynomialGCD[$poly_1$,$poly_2$]：多項式$poly_1$と$poly_2$の最大公約式を見いだす。

(2) ｓｔｕｒｍ2

引数をスツルム列の個数(一般に$f_0(x)$の次数)とし$f_0(x), f_1(x)$を基準にスツルム列を構成する。

$f_{i-1}(x)$を$f_i(x)$で割ったときの剰余項を$f_{i+1}(x)$とし帰納的に構成する。

構成したスツルム列は、Tableを用いてsturm'に格納しておく。

(プログラム上では関数列を$g_i(x)$として、スツルム列を構成している)

☆N[*expr*]：*expr*の値を数値で返す。

☆PolynomialRemainder[$poly_1$,$poly_2$,*x*]：*x*の多項式$poly_1$を$poly_2$で割り、その剰余項を見いだす。

☆Table[*expr*,{*i*,*imin*,*imax*}]：*i*=*imin*から*imax*までを通して*expr*の値のリストを作成する。

```
sturm1[stu_]:=
    h[x] = stu;
   dh[x]:= Simplify[ Dt[ h[x], x ] ];
    gcd := PolynomialGCD[ h[x], dh[x] ] ;
         g[0][x_] := h[x]/ gcd ;
         g[1][x_] := Simplify[ Dt[g[0][x],x] ];
sturm2[n_]:=
   For[ i=1 , i<n , i++ ,
      g[i+1][x_] = N[-(PolynomialRemainder[g[i-1][x],g[i][x],x])];
      sturm'     = Table[ g[i][x] , {i,0,n} ]
      ]
```

●スツルム列に数値を代入した時の符号変化回数の測定。

(1) $n_0$

数値ｘをスツルム列に代入し、その時の列を左から右に見たときの、列の符号変化の回数を調べる。

Blockで符号検査するべき列を予め評価(sig3)し固定(sign)する。Moduleで初期値を設定。

ｉは列内の位置、ｊは符号の変化回数を表す。列内のi番目とi+1番目を掛けて負の時にｊを増やす。

列のi+1番目が０になるとループを出る。sig3で０を追加しているので、無限ループになる事は無い。

☆Block[{*x*,*y*,･･･},*expr*]：記号*x*,*y*,･･･の局所値を使って*expr*を評価。

☆Module[{*x*=$x_0$,･･･},*expr*]：*x*,･･･に初期値を定義する。

☆For[*start*,*test*,*incr*,*body*]：*start*を実行、*test*がTrueを与えなくなるまで*body*と*incr*を反復評価。

☆Which[$test_1$,$value_1$,$test_2$,$value_2$,...]：それぞれの$test_i$を評価し、Trueを与える最初のものと対応する$value_i$の値を返す。

(2) ｓｉｇ1～ｓｉｇ3

スツルム列に数値ａを代入したときの符号検査。

ｱ) sig1：各スツルム列に数値ａを代入し、その符号(Sign)をとる。

ｲ) sig2：sig1の中に０がある時は、それを除く。

ｳ) sig3：sig2の最後に０を追加。

このｱ)～ｳ)の操作で、スツルム列は最後だけが０で、残り全ては１か－１である。

☆Sign[*x*]：x<0,x=0,x>0に対し、それぞれ-1,0,1を与える。

☆Delete[*expr*,*n*]：*expr*における位置*n*の要素を削除する。

☆Position[*expr*,*pattern*]：*expr*に出現する*pattern*に適合するオブジェクトの位置のリストを与える。

☆Append[*expr*,*elem*]：*expr*に*elem*を付加して与える。

```
n0[x_]:=
Block[{sign=sig3[x]},
   Module[{i=1,j=0},
         For[ i=0, sign[[i+1]]!=0, i++
          Which[sign[[i]] sign[[i+1]]<0, j++,
                True                    , j
               ]
         ];
        j
        ]
    ]
sig1[a_]:=Sign[sturm'/.x->a]
sig2[x_]:=Delete[ sig1[x],Position[sig1[x],0] ];
sig3[x_]:=Append[ sig2[x],0 ] ;
```

●解の個数判定。
(1) ｋｕｋａｎ１
区間(ａ，ｂ]内の解の個数を判定。
(2) ｋｕｋａｎ
区間(ａ，ｂ]内において、ａから順に幅ｄの区間に分け、各区間における解の個数を調べる。
解がある区間とその個数を出力。
☆While[*test*, *body*]：*test*がtrueを与えなくなるまで*body*と*incr*を反復評価。
☆If[*condition*, *t*, *f*]：*condition*の評価の結果がTrueとなる場合に*t*を、Falseとなる場合に*f*を与える。
☆Print[$expr_1$, $expr_2$, ...]：$expr_i$に続いて改行(行送り)を出力する。

```
kukan1[a_,b_]:=n0[a]-n0[b]
kukan[a_,b_,d_]:=
Module[{i=0},
       While[d(i-1)+a<b, i++;
             If[kukan1[d(i-1)+a,d i+a]>=1 ,
                Print["(",d(i-1)+a,",",d i+a,"]",
                      "  " , kukan1[d(i-1)+a,d i+a]
                     ]
               ]
             ]
      ]
```

●２分法のプログラム。

```
binary[a_,b_,eps_,binf_]:=
 For[i=0; a'=a; b'=b; sol = a'+0.5(b'-a'),
     Abs[binf/.x->sol]>eps, i++,
         sol = a'+0.5(b'-a') ;
         Which[(binf/.x->a)(binf/.x->sol)<=0,b'=sol,
                          True                ,a'=sol
               ];
    ];
```

以上のプログラムを利用し、実際にｆ(ｘ)＝０を解く事にする。計算上ではプログラムsturm1で構成した、重根が除かれた$f_0(x)$を用い$f_0(x)$＝０を解くことにする。まず、実数解を求める。
●方程式 $x+2x^2+3x^3+\cdots+8x^8=0$ に対して、スツルム列を構成する。

```
f[x_]:=Sum[i(x^i),{i,1,8}]
sturm1[f[x]];
sturm2[8]//Timing
{0.93 Second, Null}
```

実数解の存在範囲は$|x|<1+\max|a_i/a_0|$であるので、$f_0(x)$＝０の解の存在範囲は区間(－２，２]内に限られる。これはｆ(ｘ)＝０の解の存在範囲に一致する。
●区間(－２，２]で解の存在する区間を表示。(幅:0.5)

```
kukan[-2,2,0.5]//Timing
(-1.,-0.5]  1
(-0.5,0.]  1
{0.55 Second, Null}
```

●区間(－１，－０．５]と区間(－０．５，０]内に実根が１個づつあるので、
各区間においてその実根を求める。収束条件を eps＝$10^{-6}$ として解を表示。

```
binary[-1,-0.5,10^(-6),f[x]]//Timing
    sol1      = sol;
    count[1] =  i ;
binary[-0.5,0,10^(-6),f[x]]//Timing
    sol2      = sol;
    count[2] =  i ;
Print[ "Solution1 = ",sol1,"      ",count[1]," Recursion" ]
Print[ "Solution2 = ",sol2," ",count[2]," Recursion" ]
{0.11 Second, Null}
{0.11 Second, Null}
Solution1 = -0.714538      19 Recursion
                        -7
Solution2 = -9.53674 10    19 Recursion
```

●上の解をｆ(ｘ)にそれぞれ代入し誤差を表示。

```
{ Simplify[f[sol1]],Simplify[f[sol2]] }
                -7                -7
{-3.04254 10   , -9.53672 10   }
```

次に、解をｘ＝ａ＋ｂｉ(ａ,ｂは実数,ｂ≠０)と置き、虚数解を求める。

●$f_0$(ｘ)にｘ＝ａ＋ｂｉを代入し展開する。展開結果を$f_0$(ａ＋ｂｉ)＝ｕ(ａ,ｂ)＋ｂ ｖ(ａ,ｂ)ｉ
とする。これからｕ(ａ,ｂ)＝０,ｖ(ａ,ｂ)＝０とし両者からｂを消去し
ａのみの方程式(real(ａ)：解の実数部)を構成する。

(1) ｅｘｐａｎｄｆ
ｆ$_0$(ａ＋ｂｉ)を展開する。
☆**ComplexExpand[*expr*]**：全ての変数が実数である事を前提にして*expr*を展開。

(2) ｒｅｐａｒｔ
引数を関数とし、その関数の内部表記においてComplex[*0,n*]->Complex[*0,0*]
とする事で虚数部分を消去し、実数部分のみを残す。
☆**Complex[*a,b*]**：ａ＋ｂｉの内部表記

(3) ｕ,ｖ
ｱ) ｕ：ｅｘｐａｎｄｆで$f_0$(ａ＋ｂｉ)を展開、ｒｅｐａｒｔで実部のみ取り出す。
ｲ) ｖ：$f_0$(ａ＋ｂｉ)からｕ(実部)を引き、－ｉを掛けると虚部となる。
これを更にｂで割る。
ｕ,ｖともにSimplifyを用いて、最も簡単な形にしてある。

(4) ｒｅａｌ’
ｕ(ａ,ｂ)＝０,ｖ(ａ,ｂ)＝０の両者からｂを消去する。
これにより、ａのみの方程式(real(ａ)：解の実数部)が構成される。
その形式は「ｇ(ａ)＝ｃ(定数)」となっている。
☆**Eliminate[*eqns, vars*]**：連立方程式から特定の変数を消去する。

(5) ｒｅａｌ(ａ)
ｒｅａｌ’を「ｇ(ａ)－ｃ＝０」と直し、その左辺をｒｅａｌ(ａ)と置く。

```
expandf = ComplexExpand[g[0][x]/.x->a+b I];
repart[z_]:= z/.Complex[0,n_]->Complex[0,0]
u[a_,b_]:= Simplify[repart[expandf]];
v[a_,b_]:= Simplify[(expandf-u[a,b])(-I)/b];
  real'  = Eliminate[{u[a,b]==0,v[a,b]==0},b];//Timing
real[a_] = real'[[1]]-real'[[2]];
{403.59 Second, Null}
```

●実部の方程式(real)に対し、スツルムの列を構成。

```
sturm1[real[x]];
sturm2[28]//Timing
{18.95 Second, Null}
```

ここで、実部の方程式(real)の実数解の存在範囲を調べておく。

●方程式(real)の最高次数(２８次)の係数ａ$_0$を表示。
☆**Coefficient[*expr, form, n*]**：*expr*における*form*^*n*の係数を与える。

```
a0=Coefficient[real[a],a,28]
4398046511104
```

●方程式(real)の最高次(２８次)の係数を１とし、各係数ａ$_i$の絶対値が最大のものを表示。

```
Max[N[Abs[CoefficientList[real[a],a]/a0]]]//Timing
{0.05 Second, 6.19653}
```

係数|$a_i$|の中で最大の数は６．１９なので、方程式(real)の実数解は
区間[－７．２，７．２]内に限る事が分かる。

●区間(－７．２，７．２]で解の存在する区間を表示。(幅:0.5)

```
kukan[-7.2,7.2,0.5]//Timing
```

```
(-0.7,-0.2]  2
(-0.2,0.3]  1
(0.3,0.8]  1
{11.75 Second, Null}
```

●実根が区間内に１つとなるように、区間を狭める。(幅:0.1)

```
kukan[-0.7,-0.2,0.1]//Timing
```

```
(-0.6,-0.5]  1
(-0.4,-0.3]  1
{2.86 Second, Null}
```

解の実数部はそれぞれ区間(－０．６,－０．５], (－０．４,－０．３], (－０．２,０．３], (０．３,０．８]にあるので、次に各区間に対して実数部を決定し、その実数部を元に虚数部を決定する。

### Ｉ) 区間(－０．６,－０．５] のとき

●区間(－０．６,－０．５] において、方程式(real)の解を２分法で求める。
その解(実部:sol1)をｕ(ａ,ｂ)＝０,ｖ(ａ,ｂ)＝０に代入、ｕ(sol1,ｂ)＝０,ｖ(sol1,ｂ)＝０とする。
これは両方ともｂについての方程式であり、両方程式をともに満たすｂを求める。
また、complex1＝ｖ(sol1,ｂ), complex2＝ｕ(sol1,ｂ)とする。

```
binary[-0.6,-0.5,10^(-6),real[x]];//Timing
    sol1     = sol;
    count[3] =  i ;
complex1[b_]=Simplify[v[a,b]/.a->sol1];
complex2[b_]=Simplify[u[a,b]/.a->sol1];
```

```
{0.66 Second, Null}
```

まず、complex1の解の存在範囲を調べる。
●complex1に対し、スツルム列を構成する。

```
sturm1[complex1[x]];
sturm2[6]//Timing
```

```
{0.39 Second, Null}
```

●complex1の最高次(６次)の係数($b_0$)を調べる。

```
b0=Coefficient[complex1[b],b,6]
```

```
26.755
```

●complex1の最高次の係数を１とし、各係数の絶対値が最大のものを表示する。

```
Max[N[Abs[CoefficientList[complex1[b],b]/b0]]]//Timing
```

```
{0. Second, 1.45109}
```

上の結果から、complex1＝ｖ(sol1,ｂ)＝０の解の存在範囲は区間(－２．５,２．５]に限ることが分かる。
更に虚数部は必ず共役になる事から、その正根のみを求めればよい。
つまり、区間[０,２．５]で解を調べれば十分である。
●区間[０,２．５]で解の存在する区間を表示。(幅:0.3)

```
kukan[0,2.5,0.3]//Timing
```

```
(0,0.3]  1
(0.3,0.6]  1
(0.9,1.2]  1
{0.38 Second, Null}
```

同様に、complex2の解の存在範囲を調べる。
●complex2に対し、スツルム列を構成する。

```
sturm1[complex2[x]];
sturm2[8]//Timing
{0.82 Second, Null}
```

●complex2の最高次(８次)の係数($b_0$)を調べる。

```
b0=Coefficient[complex2[b],b,8]
8
```

●complex2の最高次の係数を１とし、各係数の絶対値が最大のものを表示する。

```
Max[N[Abs[CoefficientList[complex2[b],b]/b0]]]//Timing
{0.06 Second, 5.30839}
```

上の結果から、complex2＝u(sol1,b)＝０の解の存在範囲は区間(－６.４,６.４]に限ることが分かる。
先と同様の理由で、区間[０,６.４]で解を調べれば十分である。
●区間[０,６.４]で解の存在する区間を表示。(幅:0.3)

```
kukan[0,6.4,0.3]//Timing
(0.3,0.6]  1
(0.6,0.9]  1
(2.1,2.4]  1
{0.93 Second, Null}
```

complex1の解の存在範囲{(0,0.3],(0.3,0.6],(0.9,1.2]}と
complex2の解の存在範囲{(0.3,0.6],(0.6,0.9],(2.1,2.4]}を比較すると
共通根(虚部:sol2)の存在範囲が分かる。それは区間(０.３,０.６]である。
●区間(０.３,０.６] において、方程式(complex1)の解を２分法で求める。
その解が虚部(sol2)である。

```
binary[0.3,0.6,10^(-6),complex1[x]];//Timing
    sol2     = sol;
    count[4] =  i ;
Print[" Solution3 = ", sol1 + I sol2 ]
Print[" Solution4 = ", sol1 - I sol2 ]
Print[" Re_Part ",count[3]," Recursion"]
Print[" Im_Part ",count[4]," Recursion"]
{0.11 Second, Null}
 Solution3 = -0.527421 + 0.490671 I
 Solution4 = -0.527421 - 0.490671 I
 Re_Part 32 Recursion
 Im_Part 20 Recursion
```

●上の解をｆ(ｘ)に代入し誤差を表示。

```
Simplify[f[sol1+I sol2]]
          -8              -8
6.45654 10   - 9.44566 10   I
```

**Ⅱ) 区間(－0.4,－0.3] のとき**

●区間(－0.4,－0.3] において、方程式(real)の解を2分法で求める。
その解(実部:sol1)をu(a,b)＝0,v(a,b)＝0に代入する。
また、complex1＝v(sol1,b), complex2＝u(sol1,b)とする。

```
binary[-0.4,-0.3,10^(-6),real[x]];//Timing
    sol1     = sol;
    count[5] =  i ;
complex1[b_]=Simplify[v[a,b]/.a->sol1];
complex2[b_]=Simplify[u[a,b]/.a->sol1];
{0.49 Second, Null}
```

まず、complex1の解の存在範囲を調べる。

●complex1に対し、スツルム列を構成する。

```
sturm1[complex1[x]];
sturm2[6]//Timing
{0.43 Second, Null}
```

●complex1の最高次(6次)の係数($b_0$)を調べる。

```
b0=Coefficient[complex1[b],b,6]
15.8652
```

●complex1の最高次の係数を1とし、各係数の絶対値が最大のものを表示する。

```
Max[N[Abs[CoefficientList[complex1[b],b]/b0]]]//Timing
{0. Second, 1.}
```

上の結果から、complex1＝v(sol1,b)＝0の解の存在範囲は区間(－2,2]に限ることが分かる。
先と同様の理由で、区間[0,2]で解を調べれば十分である。

●区間[0,2]で解の存在する区間を表示。(幅:0.6)

```
kukan[0,2,0.6]//Timing
(0,0.6]  1
(0.6,1.2]  1
{0.22 Second, Null}
```

同様に、complex2の解の存在範囲を調べる。

●complex2に対し、スツルム列を構成する。

```
sturm1[complex2[x]];
sturm2[8]//Timing
{0.88 Second, Null}
```

●complex2の最高次(8次)の係数($b_0$)を調べる。

```
b0=Coefficient[complex2[b],b,8]
8
```

●complex2の最高次の係数を1とし、各係数の絶対値が最大のものを表示する。

```
Max[N[Abs[CoefficientList[complex2[b],b]/b0]]]//Timing
{0. Second, 2.13568}
```

上の結果から、complex2＝u(sol1,b)＝0の解の存在範囲は区間(－3.2,3.2]に限ることが分かる。
先と同様の理由で区間[0,3.2]で解を調べれば十分である。

●区間[0,3.2]で解の存在する区間を表示。(幅:0.6)

```
kukan[0,3.2,0.6]//Timing
(1.2,1.8]  1
{0.33 Second, Null}
```

complex1の解の存在範囲{(0,0.6],(0.6,1.2]}と
complex2の解の存在範囲{(1.2,1.8]}を比較すると共通根(虚部:sol2)の存在範囲が分かるのであるが、
今の場合その様な区間は存在しない。
つまり、実部が区間(－0.4,－0.3]の解のときは 『不適』と言える。

以下同様に行う。II)の様に、『不適』になることはない。

**III) 区間(−0.2,0.3] のとき**

●区間(−0.2,0.3] において、方程式(real)の解を2分法で求める。
その解(実部:sol1)をu(a,b)=0,v(a,b)=0に代入する。
complex1=v(sol1,b), complex2=u(sol1,b)とする。

```
binary[-0.2,0.3,10^(-6),real[x]];//Timing
    sol1     = sol;
    count[5] =  i ;
complex1[b_]=Simplify[v[a,b]/.a->sol1];
complex2[b_]=Simplify[u[a,b]/.a->sol1];
{0.44 Second, Null}
```

先と同様にcomplex1, complex2それぞれに対しスツルム列を構成し、解の存在区間を求める。
それらの共通区間は区間(0.7,0.8]となる。

●区間(0.7,0.8] において、方程式(com1)の解を2分法で求める。その解が虚部(sol2)である。

```
binary[0.7,0.8,10^(-6),complex1[x]];//Timing
    sol2     = sol;
    count[6] =  i ;
Print[" Solution5 = ", sol1 + I sol2 ]
Print[" Solution6 = ", sol1 - I sol2 ]
Print[" Re_Part ",count[5]," Recursion"]
Print[" Im_Part ",count[6]," Recursion"]
{0.06 Second, Null}
 Solution5 = -0.051788 + 0.738248 I
 Solution6 = -0.051788 - 0.738248 I
 Re_Part 20 Recursion
 Im_Part 17 Recursion
```

●上の解をf(x)にそれぞれ代入し誤差を表示。

```
Simplify[f[sol1+I sol2]]
            -6             -7
-1.28032 10   + 1.94124 10   I
```

**IV) 区間(0.3,0.8] のとき**

●区間(0.3,0.8] において、方程式(real)の解を2分法で求める。
その解(実部:sol1)をu(a,b)=0,v(a,b)=0に代入する。
complex1=v(sol1,b), complex2=u(sol1,b)とする。

```
binary[0.3,0.8,10^(-6),real[x]];//Timing
    sol1     = sol;
    count[7] =  i ;
complex1[b_]=Simplify[v[a,b]/.a->sol1];
complex2[b_]=Simplify[u[a,b]/.a->sol1];
{0.93 Second, Null}
```

先と同様にcomplex1, complex2それぞれに対しスツルム列を構成し、解の存在区間を求める。
それらの共通区間は区間(0.6,0.7]となる。

●区間(0.6,0.7] において、方程式(com1)の解を2分法で求める。その解が虚部(sol2)である。

```
binary[0.6,0.7,10^(-8),complex1[x]];//Timing
    sol2     = sol;
    count[8] =  i ;
Print[" Solution7 = ", sol1 + I sol2 ]
Print[" Solution8 = ", sol1 - I sol2 ]
Print[" Re_Part ",count[7]," Recursion"]
Print[" Im_Part ",count[8]," Recursion"]
{0.11 Second, Null}
 Solution7 = 0.498978 + 0.605421 I
 Solution8 = 0.498978 - 0.605421 I
 Re_Part 45 Recursion
 Im_Part 23 Recursion
```

●上の解をf(x)にそれぞれ代入し誤差を表示。

```
Simplify[f[sol1+I sol2]]
             -10             -11
4.77118 10    + 3.704 10    I
```

## 6．考察

(1) 方程式 $x+2x^2+3x^3+\cdots+8x^8=0$ とを解いたときの計算時間の比較。

| 我々の方法<br>スツルムの定理･2分法 | 所要時間<br>(秒) | 所要時間割合<br>(%) |
|---|---|---|
| 実数根の範囲限定 | 1.48 | 0.329 |
| 実数根の確定 | 0.22 | 0.049 |
| 実部の方程式(real)の算出 | 403.59 | 89.62 |
| real の範囲限定 | 33.61 | 7.462 |
| 複素根の実部確定 | 2.30 | 0.511 |
| com1, com2の共通範囲算出 | 8.88 | 1.971 |
| 複素根の虚部確定 | 0.28 | 0.062 |

| | 我々の方法 | | *Mathematica*のNSolve |
|---|---|---|---|
| 総計算時間 | 450.36 | | 0.11 |
| 時間比 | 4094 | : | 1 |

(2) 実部の方程式(real)の算出

このプログラムでの最大の短所は、実部の方程式(real)の算出に403.59(sec)もの時間を費やしてしまう点にある。実際にはu(a,b)＝0，v(a,b)＝0の２式から文字bを消去している箇所であるが、この計算にこれだけの時間を必要とするのは次の２つの理由によると思われる。

① 「結果的に実部の方程式(real)の係数が４兆、次数が28次にもなってしまっている。」

解くべき方程式 $f(x)=\sum_i^8 i x^i$ が８次方程式であるから、解の実部は最大で８個のはずである。しかし、実際には実部の方程式(real)は次数が28次にもなっている。

〔 元の方程式が $f(x)=\sum_i^n i x^i$ の時、実部の方程式(real)の次数は $n(n-1)/2$ 次になると予想される。実際、３次方程式から８次方程式まではその次数に従う。 〕

つまり、必要の無い虚数解が最大で20個も入ってしまっている事になる。実数解のみが必要であるので方程式(real)は８次方程式に留めたいものである。

何故この様に次数が上がってしまうのか関しては現在検討中である。

② 「関数Eliminateは数値による近似計算が出来ない。」

関数Eliminateはuをvで割った余りを計算し、再びuをその余りで割る、という事を繰り返していると私は予想しているのだが、仮にこの予想が正しいとすると、２つの整式の割算計算であるので、数値による近似計算が出来ない。

(3) 内部表現

プログラム上では「$f_0(a+bi)$ を展開した結果から、実数部のみを取り出す」という計算を、内部表記において Complex[0,n]→Complex[0,0] とする事で求めている。

この様に*Mathematica*には内部表現なるものが存在し、入力した数式などを内部表記に直して計算を行い、その計算結果を内部表記から普通の数式に直し出力している。

その過程はTraceというコマンドで見る事が出来る。

例えば、a＋biから実部aを取り出す過程は次の通りである。

```
zz := a + b I
repart[z_]:=z/.Complex[0,n_]->Complex[0,0]
Trace[repart[zz]]
{{zz, a + b I, {{I, I}, b I, I b}, a + I b},
 repart[a + I b], a + I b /.
  Complex[0, n_] -> Complex[0, 0],
 {{Complex[0, 0], 0}, Complex[0, n_] -> 0,
  Complex[0, n_] -> 0}, a + I b /. Complex[0, n_] -> 0,
 a + 0 b, {0 b, 0}, a + 0, 0 + a, a}
```

*Mathematica*の組み込みオブジェクトは(当然NSolveも)、その計算過程は全て内部表記で行っている。

一方自作のプログラムは、例えば「２分法」１つだけ見ても「数式→内部表記→数式」という「変換」を行っている。

万一、「組み込みオブジェクトNSolve」と「自作のプログラム」が全く同じ計算であっても変換時間分だけ、自作プログラムの方が遅くなる事は明白である。

つまり、単に計算時間だけの比較ではNSolveがどんな計算を行っているのかを知る事は出来ない様に思われる。